IN THE CLAIMS

Please amend the claims as follows:

1. (Previously Presented) A method comprising:

loading a first set of instructions into an execution unit, wherein the first set of instructions includes an unresolved reference to a second set of instructions, wherein the loading includes replacing the unresolved reference with an address of a third set of instructions;

executing instructions of the first set;

executing instructions of the third set, wherein executing instructions of the third set includes loading the second set of instructions into the execution unit; and

executing instructions of the second set.

2. (Previously Presented) The method of claim 1, wherein the first set of instructions includes an executable object module.

3. (Previously Presented) The method of claim 2, wherein the executable object module is in the Mach-O object format.

4. (Previously Presented) The method of claim 1, wherein the second set of instructions includes a separately compiled object module.

5. (Previously Presented) The method of claim 1, wherein the third set of instructions includes a loader unit.

6. (Previously Presented) The method of claim 1, wherein the loading does not include determining whether the unresolved reference refers to a defined external symbol.

7. (Previously Presented) A method comprising:

compiling a source code module into an executable object module that includes an unresolved reference to a separately compiled object module;

loading the executable object module, wherein the loading includes replacing the unresolved reference with a reference to a system module, and wherein neither the compiling nor the loading include determining whether the unresolved reference refers to a defined external symbol;

executing the executable object module, wherein the executing includes,

calling the system module for loading the separately compiled object module; and executing the separately compiled object module.

8. (Previously Presented) The method of claim 7, wherein the system module includes a loader unit.

9. (Previously Presented) The method of claim 8, wherein the loader unit is a dyld loader .

10. (Previously Presented) The method of claim 7, wherein the source code module includes instructions of a dialect of the C programming language.

11. (Previously Presented) The method of claim 7, wherein the executable object module is in the Mach-O object format.

12. (Previously Presented) A method comprising:

creating an executable object module that includes symbolic references to addresses in ones of a set of one or more separately compiled object modules, wherein the executable object module includes a page-aligned code segment and a page-aligned data segment, and wherein the object module includes resolved internal code-to-data offsets;

replacing the symbolic references with addresses to a loader subroutine;

executing the executable object module, wherein executing includes, executing the loader subroutine to load one of the separately compiled object modules; and

executing the one of the separately compiled object modules.

13. (Previously Presented) The method of claim 12, wherein the executable object module is in the Mach-O object format.

14. (Previously Presented) The method of claim 12, wherein the loader subroutine is included in a dynamic loader, and wherein the dynamic loader is dyld.

15. (Previously Presented) The method of claim 12, wherein the unresolved reference is a reference is a function call to a function included in one of the separately compiled object modules of the set.

16. (Previously Presented) The method of claim 12, wherein the unresolved reference is a reference to a variable defined within one of the separately compiled objects of the set.

17. (Previously Presented) An apparatus comprising:

a compiler unit to create an executable object module based on a source code module, wherein the executable object module includes an unresolved reference to a separately compiled object module;

a storage unit to store the executable object module;

an execution unit to receive the executable object module; and

a loader unit to find the executable object module in the storage unit and present the executable object module to the execution unit, wherein the loader unit is to replace the unresolved reference with a reference to a system module, and wherein the loader unit is not to determine whether the unresolved reference refers to a defined external object module.

18. (Previously Presented) The apparatus of claim 17, wherein the source code module includes instructions that are of a dialect of the C programming language.

19. (Previously Presented) The apparatus of claim 17, wherein the executable object module is in the Mach-O object format.

20. (Previously Presented) The apparatus of claim 17, wherein the compiler unit compiles Objective C programming language instructions.

21. (Previously Presented) An apparatus comprising:

a loader unit to load a first set of instructions into a memory unit, wherein the first set of instructions includes an unresolved reference to a second set of instructions, the loader unit to replace the unresolved reference with an address of a third set of instructions; and

an execution unit to execute instructions of the first set, the execution unit also to execute instructions of the third set to determine an address of the second set of instructions, wherein the loader unit is to use the address of the second set of instructions to load the second set of instructions into the memory unit.

22. (Previously Presented) The apparatus of claim 21, wherein the first set of instructions is an executable object module.

23. (Previously Presented) The apparatus of claim 21, wherein the executable object module is in the Mach-O object format.

24. (Previously Presented) The apparatus of claim 21, wherein the second set of instructions is a separately compiled object module.

25. (Previously Presented) The apparatus of claim 21, wherein the third set of instructions is a loader module.

26. (Previously Presented) A system comprising:

a memory unit, the memory unit including, a compiler unit to create an executable object module based on a source code module, wherein the executable object module includes a symbolic reference to a separately compiled object module; and

a loader unit to present the executable object module for execution, wherein the loader unit is to replace the symbolic reference with an address to a system module, and wherein the

loader unit is not to determine whether the symbolic reference refers to a defined external object module; and

a processor to receive the executable object module from the loader unit of the memory unit.

27. (Previously Presented) The system of claim 26, wherein the source code module includes instructions that are of a dialect of the C programming language.

28. (Previously Presented) The system of claim 26, wherein the executable object module is in the Mach-O object format.

29. (Previously Presented) The system of claim 26, wherein the compiler unit compiles C programming language instructions.

30. (Previously Presented) A machine-readable storage medium that provides instructions, which when executed by a machine, cause the machine to perform operations comprising:

loading a first set of instructions into an execution unit, wherein the first set of instructions includes an unresolved reference to a second set of instructions, wherein the loading includes replacing the unresolved reference with an address of a third set of instructions;

executing instructions of the first set;

executing instructions of the third set, wherein executing instructions of the third set includes loading instructions of the second set into the execution unit; and

executing instructions of the second set.

31. (Previously Presented) The machine-readable medium of claim 30, wherein the first set of instructions includes an executable object module.

32. (Previously Presented) The machine-readable medium of claim 31, wherein the executable object module is in the Mach-O object format.

33. (Currently Amended) The machine-readable medium of claim 30, wherein the loading does not include determining whether the unresolved reference refers to a defined external symbol.[[.]]

34. (Previously Presented) The machine-readable medium of claim 30, wherein the second set of instructions includes a separately compiled object module.

35. (Previously Presented) The machine-readable medium of claim 30, wherein the third set of instructions includes a loader unit.

36. (Previously Presented) A machine-readable storage medium that provides instructions, which when executed by a machine, cause the machine to perform operations comprising:

compiling a source code module into an executable object module that includes an unresolved reference to a separately compiled object module;

loading the executable object module, wherein the loading includes replacing the unresolved reference with a reference to a system module, and wherein neither the compiling nor the loading include determining whether the unresolved reference refers to a defined external symbol;

executing the executable object module, wherein the executing includes, calling the system module for loading the separately compiled object module; and

executing the separately compiled object module.

37. (Previously Presented) The machine-readable medium of claim 36, wherein the determining the address includes looking-up the address in a master symbol table.

38. (Previously Presented) The machine-readable medium of claim 36, wherein the source code module includes instructions of a dialect of the C programming language.

39. (Previously Presented) The machine-readable medium of claim 36, wherein the system module is a loader module.

40. (Previously Presented) The machine-readable medium of claim 36, wherein the executable object module is in the Mach-O object format.

41. (Previously Presented) A machine-readable storage medium that provides instructions, which when executed by a machine, cause the machine to perform operations comprising:

creating an executable object module that includes unresolved references to a set of one or more separately compiled object modules, wherein the executable object module includes a page-aligned code segment and a page-aligned data segment, and wherein the object module includes resolved internal code-to-data offsets;

replacing the unresolved references with references to a loader subroutine;

executing the executable object module, wherein executing includes, executing the loader subroutine to load one of the separately compiled object modules; and

executing the one of the separately compiled object modules.

42. (Previously Presented) The machine-readable medium of claim 41, wherein the executable object module is in the Mach-O object format.

43. (Previously Presented) The machine-readable medium of claim 41, wherein the loader subroutine is included in a dynamic loader, and wherein the dynamic loader is dyld.

44. (Previously Presented) The machine-readable medium of claim 41, wherein the unresolved reference is a reference is a function call to a function included in one of the separately compiled object modules of the set.

45. (Previously Presented) The machine-readable medium of claim 41, wherein the unresolved reference is a reference to a variable defined within one of the separately compiled objects of the set.

46.-51. (Cancelled)